Name(s) _____ Period _____ Date _____

Activity Guide - Magic Nine Ball

Level 1: The Magic 9 Ball Prototype

Destiny and Xochitl have invited you to join “App-ily Ever After”-- their new entertainment app empire. They plan to design, create, and publish simple, addictive, and fun apps that allow users to express their creativity.

The creation that they are prototyping is Magic 9 Ball , an app that allows users to click on the image of a 9 ball in order to receive a personalized daily fortune, a lucky color that displays a name and updates the color of the pool ball to match, and a lucky number.

The app is still in prototype stage, and they are going to need to your skills as an App Developer in order to see their project across the finish line.

🖥️ Explore the prototype in order to get a sense for how the app should be designed.

With your partner, discuss the following:
❓ What is the **purpose** of the app?
❓ Describe the **functionality** of the app. How do these design decisions fit the purpose?

Level 2: Declare and Initialize Data into Lists

This app relies on having a set of amusing, enlightening, and thoughtful fortunes & colors ready to dispense. Destiny has been working hard creating these fortunes and color combinations in order to implement into the app.

She saved all of her work into a dataset called Fortune Data that we can use to populate three different lists within the app-- one for the fortunes, one for the name of the colors, and one for the hex code that will display the color. Let's begin by helping her to declare and initialize these lists.

1️⃣ Task One: Declare the following lists--
- `fortunes`
- `luckyColorName`
- `luckyColorCode`

You can find the location in the starter code to modify by matching the number next to the task (e.g. 1️⃣ in this case)

2️⃣ Task Two: Initialize these lists with data using the `getColumn()` command

Declare / Initialize Index Variables

Since we have three different lists that contain the fortunes, the color name, and color hex code, we can create two different index variables-- one to reference the fortune elements from the list, and another to reference the color name & code. We only need one index variable for color since the name and the hex code both use the one index value to describe a common color.

Discuss with your partner:
❓ Why can't we use just one index variable for all three of the different lists?

3️⃣ Declare two index variables that will keep track of an independent index for both the `fortunes` list and the `luckyColorName` / `luckyColorCode` lists & initialize both values to zero. Name them as follows:

- `fortuneIndex`
- `colorIndex`

Level 3: Adding Button Functionality

Now that we have our initial lists and variables declared / initialized, let's add in some user interactivity.

4️⃣ Find the `onEvent` that corresponds to the *fortuneButton* element and code the following within the event:

- Set our `fortuneIndex` variable to be equal to a random value from within the fortunes list. Utilize the **Random Access Pattern** described in the previous lesson.
- Use the `setText()` command to set the value of the *fortuneText* element to the value contained within the `fortunes` list at index `fortuneIndex`

🖥️ Test your code! Click the Magic 9 Ball and see if the fortunes display and update with every user interaction.

Level 4: Adding Colors into the Mix

Now that the fortunes have been coded to display correctly, the final design elements are ready to be implemented into Magic 9 Ball.

Xochitl wants not only the app to display the name of the user's lucky color, but also to change the color of the ball to match that color. She thinks that this is the most satisfying experience for a user of the app.

In our two remaining lists, `luckyColorName` & `luckyColorCode`, there are a series of color names and associated hex code values. We will use this data in order to update the display for the user as envisioned by Xochitl.

5️⃣ Find the `setLuckyColor()` function and write code that will accomplish the following:

- Implement the **List Scrolling Pattern** so that as the user clicks the Magic 9 Ball, it increments the `colorIndex` variable. When `colorIndex` reaches the end of the list, reset its value to **0**.
- Use the `setProperty()` command to update the *background-color* property of the *bgColor* element to equal the value from the `luckyColorCode` list.
- Use the `setText()` command to display the value from the `luckyColorName` list within the

luckyColorTextBox element.

🖥️ **Test your code!** Run the program and see if the color updates within the image and in the text area properly. Make sure that `colorIndex` properly resets to 0 when the end of the list is reached.

The Cherry on Top: a Lucky Number

The last design decision to implement within the app is to display a lucky number for the user when they click the Magic 9 Ball.

6️⃣ Find the `setLuckyNumber()` function and write code that will accomplish the following:

- Declare a local variable, `luckyNumber`.
- Intialize `luckyNumber` to be equal to (`fortuneIndex` * `colorIndex`).
- Use the `setText()` command to display the value of `luckyNumber` list within the *luckyNumberTextBox* element.

🖥️ **Test your code!** Run the program and see if the lucky number displays as expected.

Level 5: A Dash of your Own Creativity

Even though Destiny did an amazing job writing fortunes and compiling colors, the thing that would set our app over the top is if you, the creator, added at least one of your own fortunes and a color value / name combo.

6️⃣ Find the `addFortuneData()` function and write code that will accomplish the following:

- Use the `appendItem()` command to add a fortune string into the fortunes list.
- Visit the Color Selector tool and choose both a hex value and the name for a color. Use `appendItem()` to add this data to their respective lists.
- The `addToFavorites()` function will be called in the `onEvent` code every time that the Favorite tab is clicked by the user.

🖥️ **Test your code!** This was the final step of our app design and creation. Use the Watcher in the bottom-right corner of the Debug Console to peek into `luckyColorName`. Is your color name added correctly?